# 第3章　TMS320LF240x的CPU功能模块和时钟模块

本章介绍CPU模块和时钟模块。

## 3.1 CPU功能模块

CPU模块包括：输入定标移位器、中央算术逻辑单元（CALU）和乘法器等。CPU模块的功能结构如图3.1

### 3.1.1 输入定标移位器

该单元将来自程序/数据存储器的16位数据调整为32位数据送到中央算术逻辑单元（CALU)。因此，输入定标移位器的16位输入与数据总线相连，　32位输出与CALU单元相连。

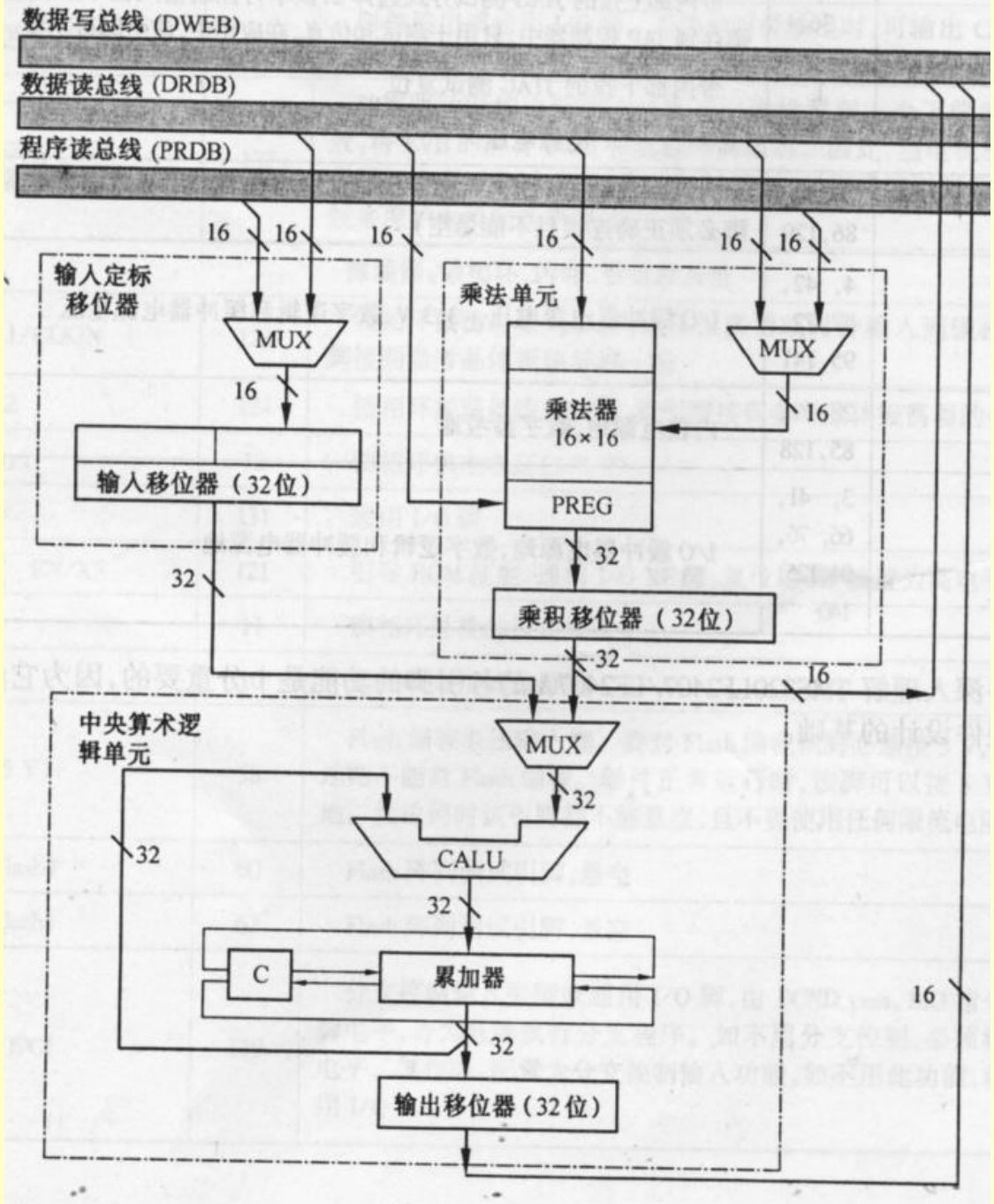


图 3.1　CPU 模块的功能结构

输入定标移位器在算术定标及逻辑操作设置时非常有用。

输入定标移位器对输入数据进行0-15位左移。左移时，输出的最低有效位（LSB）为0，最高有效位（MSB）根据状态寄存器ST1（见3.1.6）的SXM位（符号扩展方式）的值来决定是否进行符号扩展。当SXM＝1时，则高位进行符号扩展；当SXM＝0时，则高位填0。

移位的次数由包含在指令中的常量或临时寄存器（TREG）中的值来指定。

## 3.1.2 乘法器

16×16位的硬件乘法器，单个机器周期内产生一个32位的有符号或无符号乘积。

除了执行无符号乘法指令（MPYU）外，所有的乘法指令均执行有符号的乘法操作，即相乘的两个数都作为二进制的补码数，而运算结果为一个32位的二进制的补码数。

乘法器接收的两个乘数，一个来自16位的临时寄存器（TREG），另一个通过数据读总线（DRDB）取自数据存储器，或通过程序读总线（PRDB）取自程序

存储器。

两个输入值相乘后，32位的乘积结果保存在32位的乘积寄存器（PREG）中。

PREG的输出连接到乘积定标移位器，通过乘积定标移位器，乘积结果可以从PREG传到CALU或数据存储器。乘积定标移位器对乘积采用4种乘积移位方式，如表3.1所示。

移位方式由状态寄存器ST1的乘积移位方式位（PM）指定，对于执行乘法/累加操作、进行小数运算或者进行小数乘积的调整都很有用。

表3.1　　乘积定标移位器的乘积移位方式

| PM | 移位 | 作用和意义 |
| --- | --- | --- |
| 00 | 无移位 | 乘积送CALU或数据写总线，不移位 |
| 01 | 左移1位 | 移去二进制补码乘法产生的额外符号位，产生Q31格式的乘积 |
| 10 | 左移4位 | 当与一个13位的常数相乘时，移去在16×13位（常数）二进制补码产生的额外的4位符号位，产生Q31格式的乘积 |
| 11 | 右移6位 | 对乘积结果定标，以使得运行128次的乘积累加而累加器不会溢出 |

## 3.1.3 中央算术逻辑单元

中央算术逻辑单元（CALU)实现大部分算术和逻辑运算功能，大多数功能只需一个时钟周期，这些运算功能包括：16位加、16位减、布尔运算、位测试以及移位和循环功能。

由于CALU可以执行布尔运算，因此使得控制器具有位操作功能。CALU的移位和循环在累加器中完成。CALU是一个独立的算术单元，它和后面介绍的辅助寄存器算术单元（ARAU）在程序执行时，是完全不同的两个模块。

一旦操作在CALU中被执行，运算结果会被传送到累加器中，在累加器中再实现如移位等附加操作。

CALU有两个输入，一个由累加器提供，另一个由乘积寄存器（PREG）或数据定标移位器的输出提供。当CALU执行完一次操作后将结果送至32位累加器，由累加器对其结果进行移位。累加器的输出送到32位输出数据定标移位器

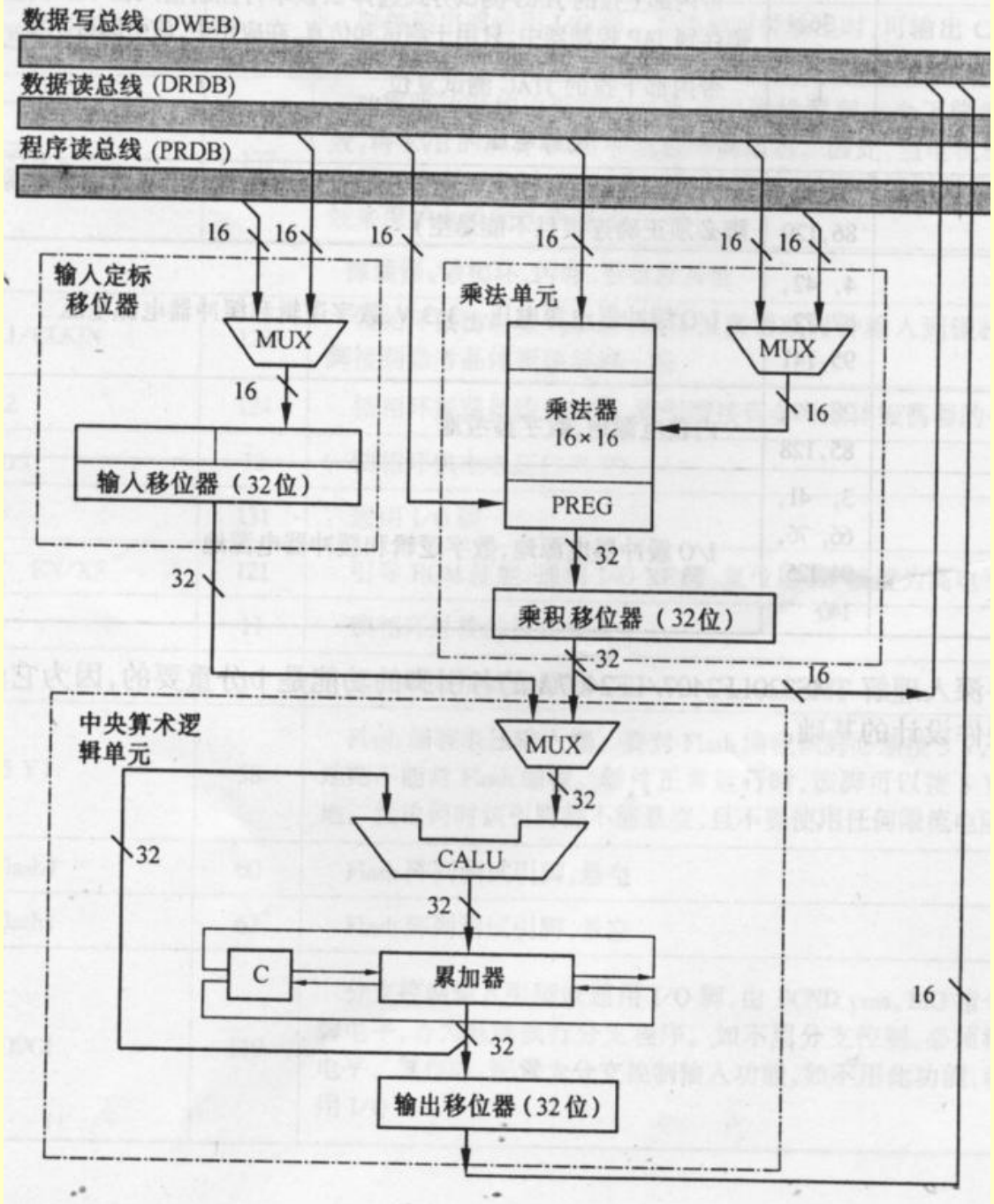


图 3.1　CPU 模块的功能结构

经过输出数据定标移位器，累加器的高、低16位字可分别被移位或存入数据寄存器。

CALU的溢出饱和方式可以由状态寄存器STO（见3.1.6小节的介绍）的溢出模式（OVM）位来使能或禁止。

根据CALU和累加器的状态，CALU可执行各种分支指令。这些指令可以根据这些状态位有意义的结合，

有条件地执行。为了溢出管理，这些条件包括OV(根据溢出跳转)和EQ(根据累加器是否为0跳转)等。另外，BACC（跳到累加器的地址）指令可以跳转到由累加器所指定的地址；不影响累加器的位测试指令（BIT和BITT）允许对数据存储器中的一个指定位进行测试。

对绝大多数的指令，状态寄存器ST1的第10位符号扩展位（SXM）决定了在CALU计算时是否使用符号扩展。

SXM=0，符号扩展无效，

SXM=1，符号扩展有效。

## 3.1.4 累加器（ACC)

当CALU中的运算完成后，其结果就被送至累加器，并在累加器中执行单一的移位或循环操作。

累加器的高位和低位字中的任意一个可以被送至输出数据定标移位器，在此定标移位后，再保存于数据存储器。与累加器有关的状态位和转移指令：

### 1．进位标志位C

ST1的第9位。下述情况之一将影响进位标志位C。

(1)加到累加器或从累加器减

- 当C=0，减结果产生借位时或加结果未产生进位时。
- 当C=1，加结果产生进位时或减结果未产生借位时。

(2)将累加器数值移1位或循环移1位

在左环移或循环左移的过程中，累加器的最高有效位被送至C位。在右环移或循环右移的过程中，累加器的最低有效位被送至C位。

## 2. 溢出方式标志位OVM

STO的第11位。OVM位决定Acc如何反映算术运算的溢出。

OVM＝1，Acc运算溢出， 累加器被设定为下列两个特定值之一：

- 若正溢出，Acc中填最大正数:7FFF FFFFh
- 若负溢出，Acc中填最大负数:8000 0000h

OVM＝0，ACC中的结果正常溢出。

3．溢出标志位OV

ST0的第12位，

C＝0，累加器未溢出；

C＝1，累加器溢出，且被锁存。

4．测试/控制标志位TC

ST1的第11位，根据被测位的值置1或清0。

与累加器有关的转移指令大都取决于C、OV、TC的状态和累加器的值。

### 3.1.5 输出数据定标移位器

它存储指令中指定的位数，将累加器输出的内容左移0—7位，然后将移位器的高位字或低位字存到数据存储器中（用SACH或SACL指令）。在此过程中，累加器的内容保持不变。

### 3.1.6 状态寄存器ST0和ST1

ST0和ST1包含了DSP运行时的各种状态和控制位。内容可被读出并保存到数据存储器（用SST指令），或从数据存储器读出加载到ST0和ST1 （用LST指令），用来在子程序调用或进入中断时实现CPU各种状态的

保存。

可用指令对ST0和ST1中的各个位单独置1或清0（SETC或CLRC指令）。

ST0各位的含义如下：

ARP（位15-13）：辅助寄存器（AR）间接寻址的指针，选择当前的8个辅助寄存器AR中的一个。AR被装载时，原ARP的值被复制到ARB中。

OV（位12）：溢出标志位。用以指示CALU中是否发生溢出，如溢出则该位为1。

OVM （位11）：溢出方式标志位

＝0，累加器中结果正常溢出。

＝1，根据溢出的情况，累加器被设定为它的最大正值或负值。

INTM（位9）：中断总开关位

＝1，所有可屏蔽中断被禁止

＝0，所有可屏蔽中断有效。

DP（位8-0）：数据存储器页面指针

9位的DP与指令中的7位形成16位的数据存储器的直接地址。

ST1各位的含义如下：

ARB（位15-13）：辅助寄存器指针缓冲器。

当ARP被加载到ST1时，原来的ARP被复制到ARB中，也可将ARB复制到ARP中。

CNF（位12）：片内DARAM配置位

0—片内DARAM映射到数据存储器区；

1—片内DARAM映射到程序存储器区。

TC（位11）：测试/控制标志位。根据被测试位的值，该位被置1或清0。

SXM（位10）：符号扩展方式位，决定在计算时是否使用符号扩展：

1—数据通过定标移位器传送到累加器时将产生符号扩展；

0—不产生符号扩展。

C（位9）：进位标志位

XF（位4）：XF引脚状态位

可用指令置1或清0。

PM（位1-0 ）：乘积移位方式

00－乘法器的32位乘积不移位，直接入CALU。

01－PREG左移1位后装入CALU，最低位填0；

10－PREG左移4位后装入CALU，低4位填0；

11－PREG输出进行符号位扩展，右移6位。

### 3.1.7 辅助寄存器算术单元（ARAU）

ARAU完全独立于中央算术逻辑单元，见图3.4。

主要功能：是在CALU操作的同时执行8个辅助寄存器AR7-AR0中的算术运算，AR7-AR0提供了强大而灵活的间接寻址能力。

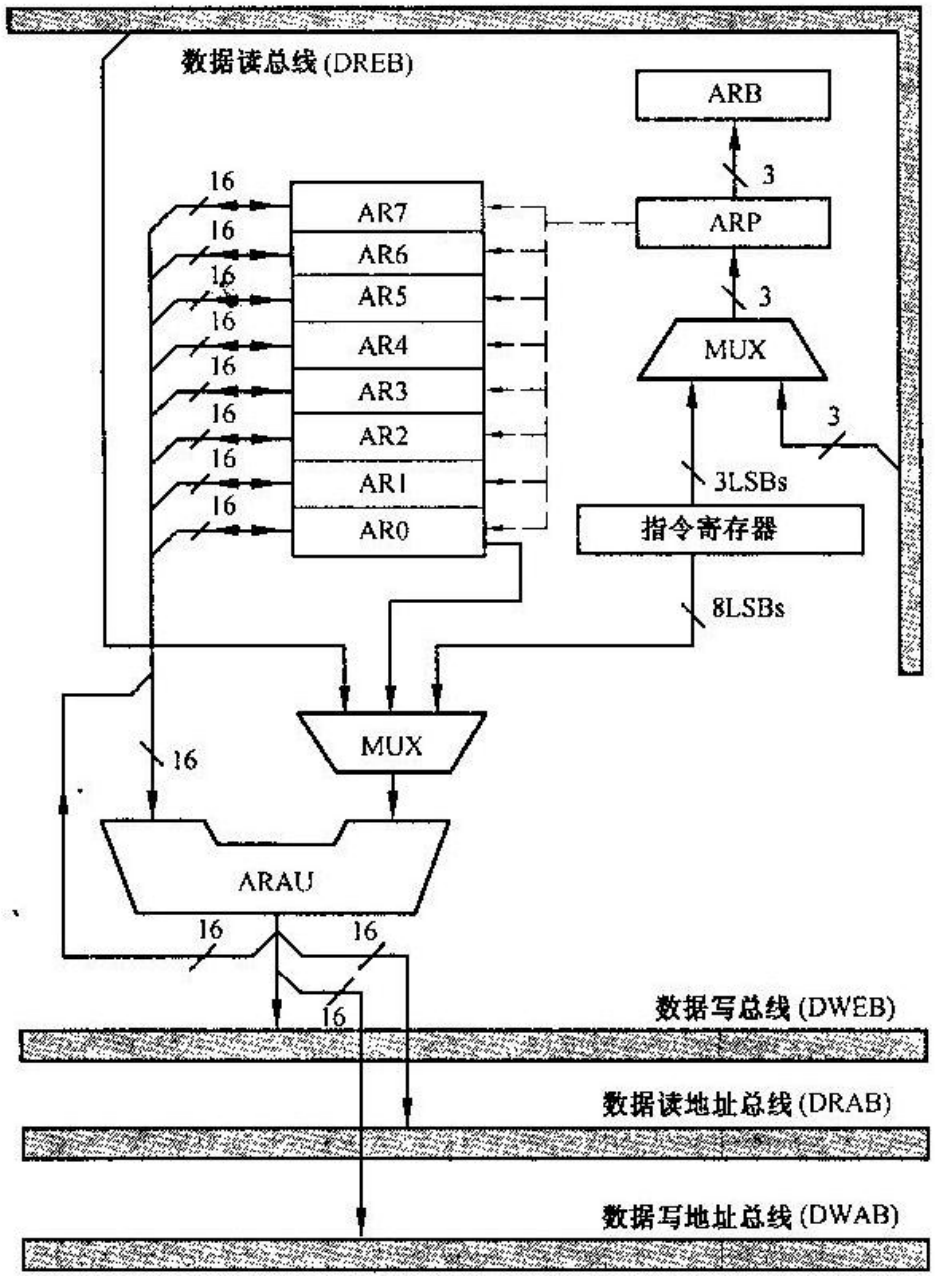


图 3.4　辅助寄存器算术单元(ARAU)

利用ARAU中的16位地址可访问数据存储器64K字空间的任一单元。

如何来选择某一辅助寄存器？可通过指令向STO的ARP指针装入3位（0－7）数据。

ARAU的8个辅助寄存器提供了强大而灵活的间接寻址能力。利用辅助寄存器中的16位地址可访问数据存储器64K字空间的任一单元。

ARAU除可对数据存储器的寻址外，还可用作它用：

（1）通过CMPR指令，利用辅助寄存器支持条件转移、调用和返回；

（2）利用辅助寄存器作为暂存单元；

（3）利用辅助寄存器进行软件计数。根据需要将其加1或减1。

## 3.2 锁相环（PLL）时钟模块和低功耗模式

LF240xDSP片内集成有锁相环(PLL)电路。可从一个较低频率的外部时钟合成片内较高工作频率的时钟。

LF240xDSP有3个引脚与时钟模块有关：

（1）XTAL1/CLKIN：外接的基准晶体到片内振荡器输入引脚；如使用外部振荡器，外部振荡器的输出必须接到该引脚。

（2）XTAL2：片内PLL振荡器驱动外部晶振的时钟输出引脚；

（3）CLKOUT/IOPE0：时钟输出或通用I/O脚。CLKOUT可用来输出CPU时钟或看门狗定时器时钟，这由系统控制状态寄存器（SCSR1，见第4章)中的位14（CLKSRC）决定。

当该脚不用于时钟输出时，就可作通用I/O。

复位时，配置为时钟输出CLKOUT。

### 3.2.1 锁相环（PLL）

PLL模块为片内所有功能模块提供必要的时钟信号，还可控制低功耗操作。

PLL支持从0.5~4倍输入时钟频率的倍率，由系统控制状态寄存器（SCSR1）的位11~9来决定。如表3.2所示。

# 表3.2　锁相环(PLL)的倍率选择

| CLKPS2 | CLKPS1 | CLKPS0 | 倍频系数 |
|---|---|---|---|
| 0 | 0 | 0 | 4 |
| 0 | 0 | 1 | 2 |
| 0 | 1 | 0 | 1.33 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0.8 |
| 1 | 0 | 1 | 0.66 |
| 1 | 1 | 0 | 0.57 |
| 1 | 1 | 1 | 0.5 |

# 1. 锁相环的时钟模块电路

锁相环的时钟模块电路如图3.5所示。

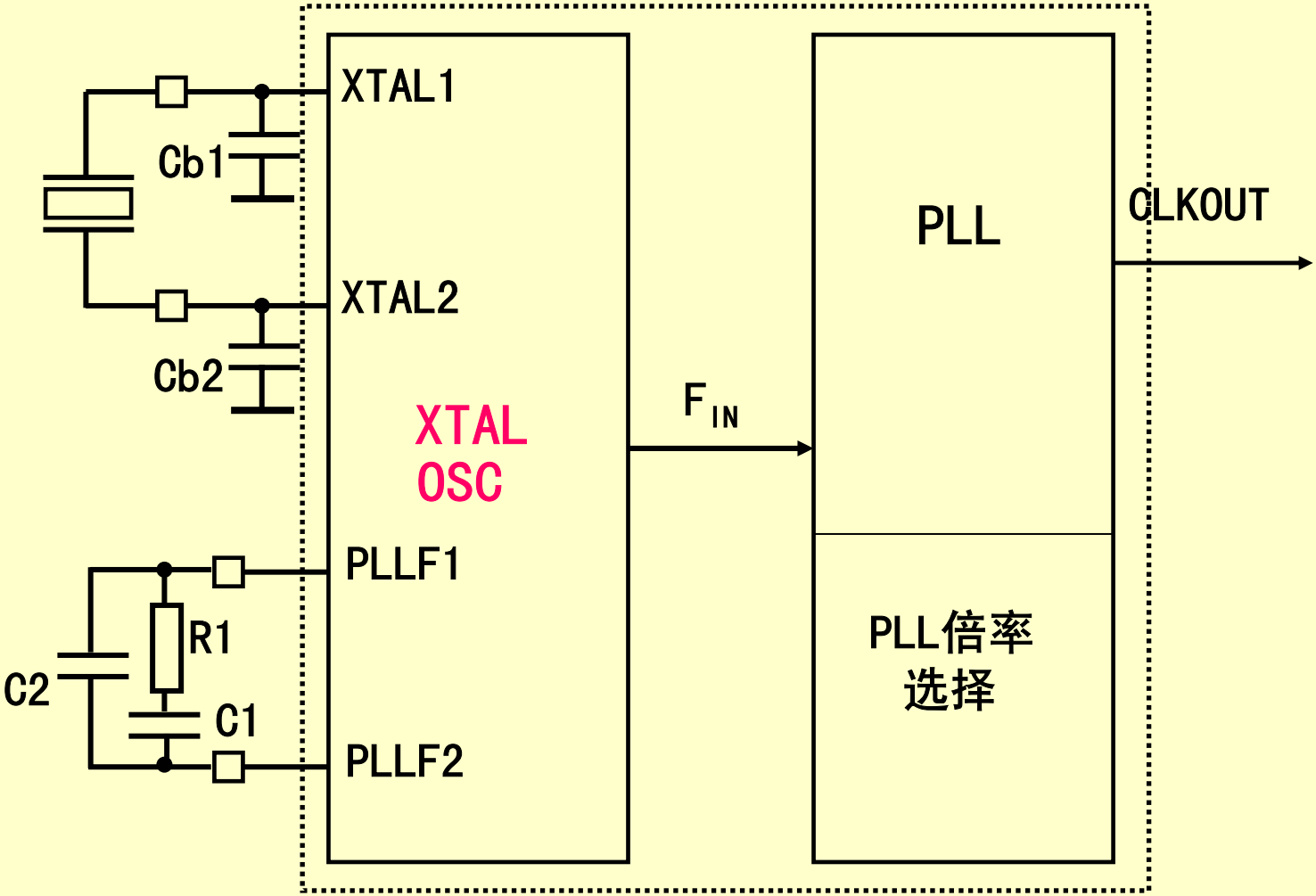


图3.5

两种时钟工作方式：

（1）内部时钟：外接基准晶体+片内PLL(锁相环)电路共同组成系统时钟电路。

（2）外部时钟：一个独立的外部时钟接至XTAL1/CLKIN引脚，此时内部时钟振荡器被旁路。

2. 外部滤波器电路回路

被PLL用来抑制信号抖动和电磁干扰，使其最小。由于电路中存在大量噪声，如何使得滤波效果最好，在设计时，需通过实验来确定滤波器回路元件的参数

滤波器电路回路接到LF240x的PLLF和PLLF2引脚（见图3.5）

滤波器电路回路元件为R1、C1和C2，电容C1和C2必须是无极性的。在不同振荡器频率下（加到XTAL1引脚的时钟频率下的R1、C1和C2的推荐值见表3.3。

表3.3　具有阻尼系数2.0的滤波元件推荐参数（P27）

所有连接PLL的PCB导线尽可能短。一个旁路电容（0.01-0.1μF）应该连接在PLLVCCA和 VSS引脚之间。如图3.6所示为在PLLVCCA和 VSS引脚

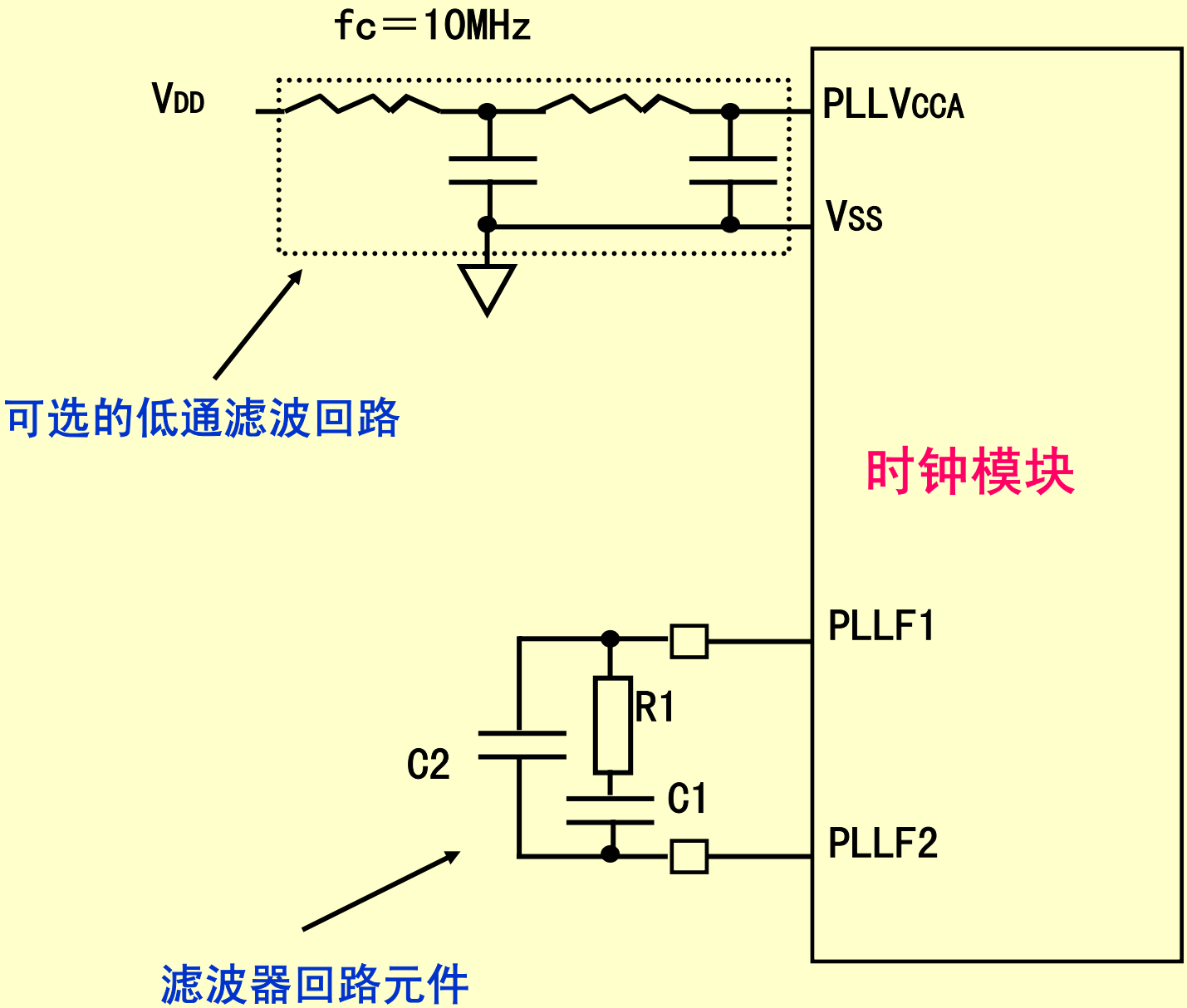
fc＝10MHz
VDD
PLLVCCA
VSS
可选的低通滤波回路
时钟模块
PLLF1
R1
C2
C1
PLLF2
滤波器回路元件

图3.6

之间增加了一个可选的低通滤波电路。

连接PLL引脚时，应注意以下几个方面：

（1）使用短引线连接PLLVCCA引脚到低通滤波器。10MHz的截断滤波器不是必需的，但是可以提高信号的抖动性能，并减少电磁干扰。

（2）使导线即可能短，保证Cbypass （0.01μF-0.1μF的陶瓷电容）最近连接于PLLVCCA和 VSS之间。

（3）使这些导线、芯片和旁路电容形成的环路面积最小。面积越大，则电磁干扰越大。要避免附近具有噪声的导线连接到时钟模块的引脚上。

## 3. 内部时钟

外接基准晶体+片内的内部时钟电路，如图3.7所示

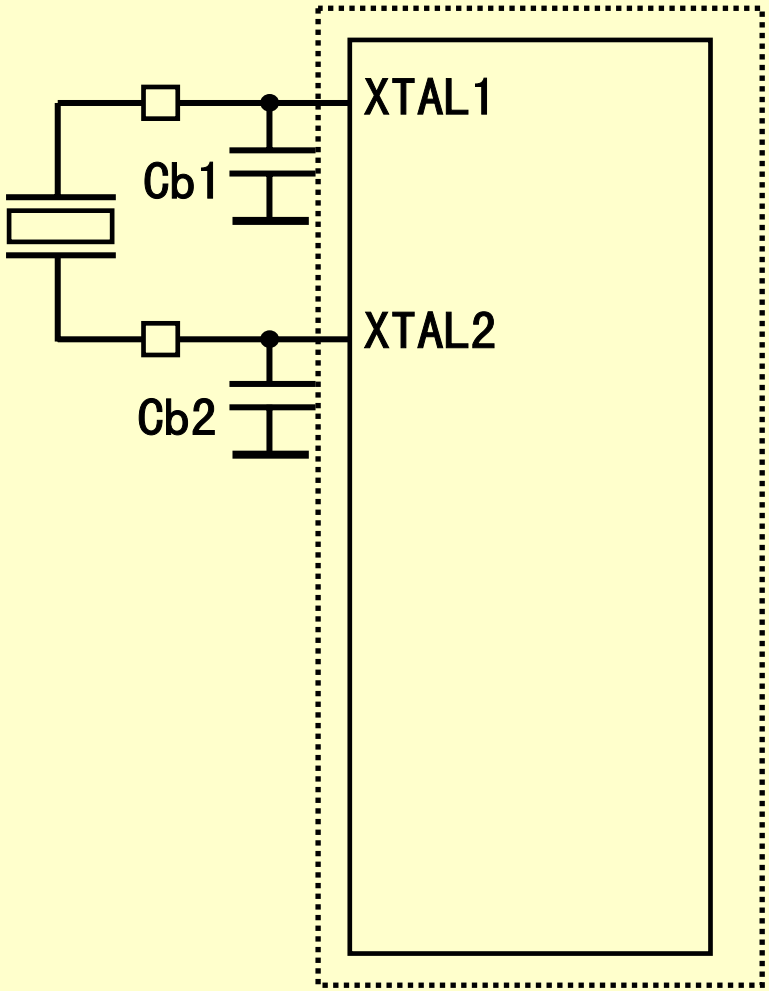


图3.7

## 4. 外部振荡器时钟

外部时钟的电路如图3.8所示。

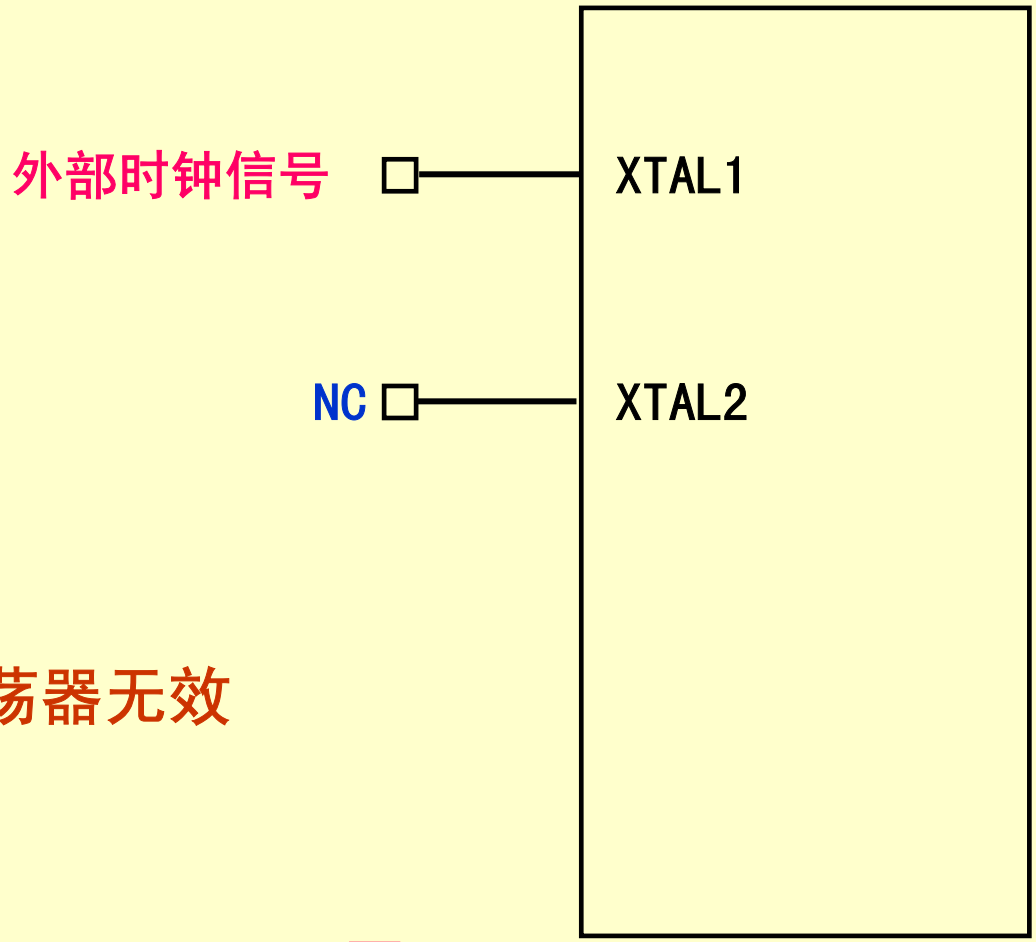


图3.8

## 5. PLL旁路方式

可设置为对片内PLL旁路的工作方式，通过复位时拉低TRST、TMS和TMS2引脚来实现。

在这种方式下，不但可以实现PLL旁路，而且可以实现PLL时钟预定标。在这种工作方式下，改变寄存器SCSR1的位11-9无效。此时改变系统时钟的唯一方法是改变输入时钟频率，系统的时钟与外输入时钟相同。例如，要获得一个30MHzCPU时钟速度，那么一个30MHz时钟CLKIN必须提供。在这种方式下，外部的滤波器元件是不需要的。

PLL旁路方式下的时钟规范如下：

（1）使用内部时钟方式，那么最小和最大的CLKIN频率分别为4MHz和20MHz。

（2）使用外部时钟方式，那么最小和最大的CLKIN频率分别为4MHz和30MHz（对2407A为40MHz）。

### 3.2.2 看门狗定时器时钟

WDCLK被用来给看门狗提供时钟源。当CPU的时钟频率为：CPUCLK(*CLKOUT*)=40MHz时，WDCLK有一个78125Hz的名义频率。

$$WDCLK = CLKOUT/512$$

WDCLK来自于CPU的CLKOUT（见图3.3）。这可以保证即使当CPU处于IDLE1或IDLE2模式（低功耗模式，见3.2.3）看门狗定时器也能持续计数。

当CPU挂起时，WDCLK将被停止。

### 3.2.3 低功耗模式

LF240x的IDLE（睡眠）指令，可关闭CPU时钟，进入睡眠状态，节约能耗。

CPU如何退出睡眠状态:收到一个中断请求或复位时。

1. 时钟域

LF240x有两个时钟域：

（1）CPU时钟域:包含大部分CPU逻辑的时钟；

（2）系统时钟域：包含外设时钟（来自CLKOUT分频）和用于CPU中断逻辑的时钟。

IDLE1模式：当CPU进入睡眠状态，CPU时钟域停止，系统时钟域继续运行。

IDLE2模式：当CPU进入睡眠状态，CPU时钟域和系统时钟域均停止，进一步降低功耗。

HALT模式：振荡器（即输入到PLL的时钟）和WDCLK被关闭。

当执行IDLE指令时，系统控制状态寄存器（SCSR1）的13、12位指明进入哪一种低功耗模式。具体如下：

0　0　—CPU进入IDLE1模式

0　1　—CPU进入IDLE2模式

1　x　—CPU进入HALT模式

## 2. 唤醒低功耗模式

### （1）复位

复位信号可使器件退出IDLE模式。

### （2）外部中断

外部中断XINTx可使器件退出低功耗模式，但不能退出HALT模式。

### （3）唤醒中断

有些外设具有启动器件时钟的能力，然后产生一个中断去响应一定的外部事件。例1：通信线路上的动作。例2：即使没有时钟运行，CAN唤醒中断也可以声明一个CAN错误中断请求

## 3. 退出低功耗模式

外设中断可以用来唤醒处于低功耗模式工作的器件，立即退出低功耗模式。

根据以下几种情况执行唤醒动作（和随后的器件动作）：

- 请求的外设中断是否使能于外设级。
- 与请求的外设中断相关的IMR.n位是否已经被使能。
- STO寄存器INTM位的状态。

## 3.2.4 片内Flash的断电与上电

进入HALT模式之前，片内Flash模块可以被断电，会使电流消耗降到最低。下面为Flash模块断电的程序。

```
; Flash模块断电的程序
LDP #0h                 ; 设置DP=0
SPLK #0008h, 60h        ; 将0008h写入Flash控制方式寄存器,
OUT 60h, #0FF0Fh        ; 将Flash置于断电模式
LACL #0h                ; 0000h为管道控制寄存器的地址
TBLW 60h                ; 写操作可以将Flash断电
```

使用PDPINTx*和RS*信号可以退出LPM2（HALT）模式。

如果PDPINTx*被用于给Flash模块上电（退出LPM2状态），当RS*自动给Flash上电，需要执行下面程序：

```
; Flash模块上电程序
   LDP #0h                  ; 设置DP=0
   SPLK #0000h, 60h         ; 设置0000h，即可将Flash退出断
                            ; 电模式
   OUT 60h, #0FF0Fh         ; 将Flash置于控制寄存器访问模式
   LACL #0h                 ; 0000h 为管道控制寄存器的地址
   TBLW 60h                 ; 写操作可以将Flash上电
   IN 60h, #0FF0Fh          ; 将Flash置于阵列访问模式
```

Flash上电后，读一个Flash中的确定地址确保Flash为应用程序的使用准备好。

例如，程序存储器中的0000h有一个操作码为7980h的“branch”指令，因此地址0000h可以读取并为7980h选中，使Flash上电有效。

《结束》